Future Generation Computer Systems 104 (2020) 212–224

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Adaptive sliding windows for improved estimation of data center resource utilization

Shuja-ur-Rehman Baig [a,b,c,*], Waheed Iqbal [c], Josep Lluis Berral [a,b], David Carrera [a,b]

[a] Universitat Politècnica de Catalunya (UPC), Spain
[b] Barcelona Supercomputing Center (BSC), Spain
[c] University of the Punjab (PU), Pakistan

ARTICLE INFO

Article history:
Received 8 April 2019
Received in revised form 11 September 2019
Accepted 27 October 2019
Available online 5 November 2019

Keywords:
Sliding windows
Adaptive observation window
Time series
Resource estimation
Data center

ABSTRACT

Accurate prediction of data center resource utilization is required for capacity planning, job scheduling, energy saving, workload placement, and load balancing to utilize the resources efficiently. However, accurately predicting those resources is challenging due to dynamic workloads, heterogeneous infrastructures, and multi-tenant co-hosted applications. Existing prediction methods use fixed size observation windows which cannot produce accurate results because of not being adaptively adjusted to capture local trends in the most recent data. Therefore, those methods train on large fixed sliding windows using an irrelevant large number of observations yielding to inaccurate estimations or fall for inaccuracy due to degradation of estimations with short windows on quick changing trends. In this paper we propose a deep learning-based adaptive window size selection method, dynamically limiting the sliding window size to capture the trend for the latest resource utilization, then build an estimation model for each trend period. We evaluate the proposed method against multiple baseline and state-of-the-art methods, using real data-center workload data sets. The experimental evaluation shows that the proposed solution outperforms those state-of-the-art approaches and yields 16 to 54% improved prediction accuracy compared to the baseline methods.

1. Introduction

Cloud computing and virtualization technologies have helped to extend the access of data center resources to users and companies globally over the Internet. Specifically, pay-as-you-go and on-demand resource provisioning features of cloud computing provide users data center resources quickly and cost-effectively. Moreover, most of the essentials for a better human life including energy, telecommunication, transportation, and health care facilities also depend on computing processes and resources deployed on data-centers. Efficient management of those resources is vital to maximizing the performance of the facility with minimal and affordable operational costs.

Data center's resource utilization forecasting is important for various reasons including resource management [1–5], energy saving [6–9], cost prediction and consolidation of virtual machines (VMs) [10–12], and capacity planning [13,14]. An accurate estimation of resource demands can greatly help to optimize the operational cost of the applications for the end uses, also for data center managers to reduce the aggregated costs of offering those resources to large numbers of users. This is found in scenarios where data center users can dynamically tune resources to minimize their usage while maintaining good quality of service levels [15], whereas the providers can increase the profit by maximizing the use of available resources [16]. Moreover, future estimation of resource utilization is helpful in other domains such as the Internet of Things [17], Cyber–Physical Systems [2], and Industry informatics [18].

Data center resources usage is highly volatile due to dynamically changing workloads, running on heterogeneous infrastructures, making it difficult to estimate future resource demands with acceptable accuracy. The problem becomes even harder due to multi-tenancy, virtualization, and co-hosted applications running on those data centers. Moreover, the knowledge about the apps running in the data center is also minimal, as usually resource providers avoid inspecting running tasks and stored data to preserve users privacy.

There have been several efforts to build estimation methods for cloud data center resource utilization using machine learning methods [19–23]. All of these methods use a fixed size sliding observation windows to train the estimation models at each step. These fixed sliding observation windows approach cannot appropriately limit the data points to capture local trends in the

* Corresponding author at: Universitat Politècnica de Catalunya (UPC), Spain.
E-mail addresses: shuja.baig@bsc.es (S.-u.-R. Baig), waheed.iqbal@pucit.edu.pk (W. Iqbal), josep.berral@bsc.es (J.L. Berral), david.carrera@bsc.es (D. Carrera).

https://doi.org/10.1016/j.future.2019.10.026

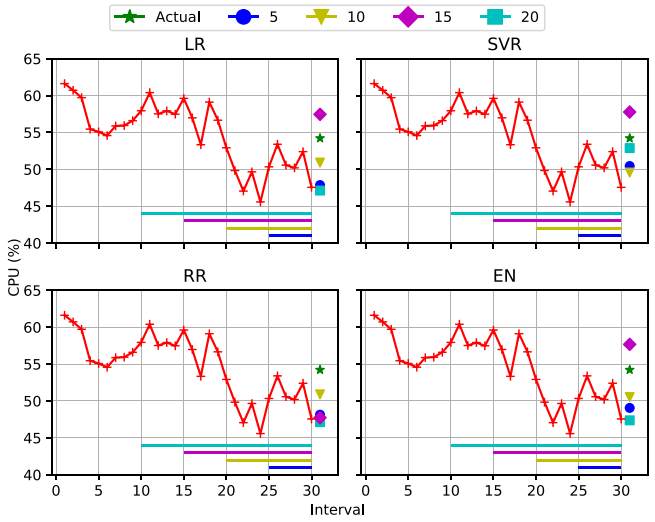


Fig. 1. CPU estimation using different size of sliding windows for various machine learning predictors.

data which can help to build estimation models with maximum prediction accuracy, as applications may not present a constant length behavior in time to fix that window size. However, if we can devise a method to adaptively select the sliding window size that can choose the appropriate recent observations, and capture the latest trend in the data, then the estimator can be trained effectively on this data to maximize their accuracy.

To show the motivation for using an appropriate observation window size for resource estimation, we conducted a preliminary experiment using Alibaba cluster traces [24], randomly selecting a virtual machine (VM) and using its CPU utilization to study the effect of different observation window sizes using different estimation methods. Fig. 1 shows the CPU utilization estimation for 31st-time interval (from a 30-time observation series) using Linear Regression (LR), Polynomial Regression (PR), Support Vector Regression (SVR), and Elastic Net (EN) for observation window sizes 5, 10, 15, and 20. We observed that window size 10 and 15 yield best estimations using LR, window size 20 gives the minimum estimation error using SVR, RR gives best estimation result for window size 10, and EN yields minimum estimation size for window size 15. These results show that the appropriate observation window size can play an important role to minimize the estimation accuracy for different estimation method. We advocate that any estimation algorithm can improve the estimations using appropriate observation window size. The problem to automatically identify the appropriate window size at every estimation step to train the model is challenging.

To further elaborate on the advantages of using adaptive sliding windows for resource estimation, we study the effect of estimation performance using different fixed sliding window sizes and adaptive observation windows. We used LR as estimation algorithm and used the entire CPU utilization data of the selected VM as a time series. We identified the optimal observation window size using exhaustive search among window sizes 2 to 30 and chosen the window size which yields maximum accuracy as the optimal window at every estimation step. Fig. 2(a) shows the mean squared error (MSE) for static sliding windows of sizes 2, 5, 10, 20, 30, and optimal window size using adaptively in each estimation step. Fig. 2(b) shows the box plot for absolute errors obtained for each estimation step for different fixed sliding window sizes and optimal window sizes identified adaptively. Fig. 2(c) shows adaptive the sliding window sizes identified adaptively at each estimation step. We observed that the adaptive observation windows yield significantly minimum estimation error comparing to static fixed size sliding windows. However, it is challenging to determine the appropriate observation window size efficiently at each estimation step for maximizing prediction accuracy.

In this paper, we address this challenging problem and propose a method using deep learning approach to adaptively select the best observation window to minimize the estimation error. The proposed adaptive sliding window identification method limits the observation window which can be used by any estimation algorithm to train the model for predicting next interval observations. We have evaluated the proposed method with the commonly used machine learning namely Linear Regression, Ridge Regression, Least Absolute Shrinkage and Selection Operator (LASSO), Elastic Net, Non-negative Least Square, and Support Vector Regression for multiple publicly available data sets. The proposed adaptive sliding windows outperform the fixed sliding windows using all estimation methods by significant margins for all data sets. We noted the maximum improvement of 40% by the proposed method. The proposed solution required sufficient history covering different workload patterns for identification of observation window sizes with higher accuracy. Most of the data center maintains a long history for resource utilization; therefore, this condition can be addressed easily.

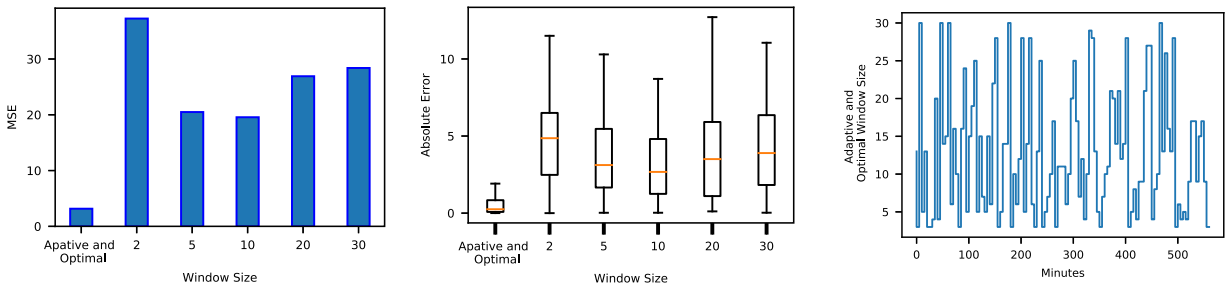
The main contributions of this paper are as follows:

- Propose a novel method to dynamically identify the best sliding window size to train the prediction model for estimating data center resource utilization.
- Evaluation and comparison of the proposed method with different baseline methods, currently used in the state-of-the-art, as candidate methods for window size estimation, aside of validation for the presented approach.
- Extensive experimental evaluation using three publicly available data sets of different real data centers.

The rest of the paper is organized as follows. Related work is presented in Section 2. The overall proposed system overview is explained in Section 3. The proposed adaptive window size predictor is presented in Section 4. Different estimation methods used for resource utilization prediction are briefly explained in Section 5. Experimental evaluation and results are provided in Sections 6 and 7 respectively. Finally, conclusion and future work are discussed in Section 8.

2. Related work

Many researchers have recently addressed data center resource estimation. For example, a recent work by Mason et al. [25] estimated the host CPU utilization using Neural Networks trained on a fixed observation window sizes containing entire day data. Zhang et al. [18] proposed cloud workload prediction system for industry informatics based on stacked autoencoders. They use a canonical polyadic decomposition format to reduce the training time by compressing the input parameters. The author also used a fixed observation window size to train and test the proposed estimation method. Another work by Nikravesh et al. [20] proposed a predictive auto-scaling system to scale the cloud resources automatically. The proposed approach uses SVM and neural network for the different type of workloads and estimation methods are trained using a fixed observation window size. However, the work studies the effect of different observation window sizes. Yang et al. [21] used linear regression to predict the cloud workload using a fixed observation window size of 4. Davis et al. [22] proposed hardware failure prediction system for cloud data centers using Neural Networks however



(a) Comparison of MSE (b) Box plot for absolute errors (c) Optimal window sizes

Fig. 2. Evaluation of fixed sliding windows of different sizes and adaptive optimal sliding windows for estimating data center CPU resources.

they also used fixed length sliding window. All of the above-mentioned works use fixed observation windows. However, there are some contributions which use dynamic observation windows for resource estimations. For example, Dalmazo et al. [26] proposed a method to us dynamic observation windows to estimate the network traffic in the cloud using statistical techniques. The proposed approach uses the variance of the data for the current and the last observation windows and adjusts the next observation window size. Klinkenberg et al. [27] use the adaptive sliding window to forecast the time series by identifying the changes in the underlying data generation process. They started with a default initial window size and kept increasing or decreasing it minimize the forecasting error. However, the proposed approach requires an extensive search to identify appropriate window size which is not feasible for real-time processing. A recent work by Tschumitschew et al. [28] select the optimal window size for the regression problem to predict the next data point in the time series based on the ratio of drift and noise. They use larger window size if noise is stronger than drift otherwise shorter window size is used for the estimation. The dynamic observation windows are used for a variety of different problems including physical activity recognition, industrial process optimization, mining frequent itemsets, wireless networks, health care, and renewable power generation. Wang et al. [29] use adaptive time series windows with Convolutional Neural Networks to optimize the industrial process operations. Authors proposed to select different time-series windows according to the steady and unsteady states in the given historical time series observations. Ouyang et al. [30,31] use optimal window size for wind power ramp prediction by minimizing the non-ramp data in time windows. Some of the recent work shows the use of a dynamic sliding window in the health care domain. For example, Smrithy et al. [32,33] use a dynamic sliding window with weighted moving averages to detect the anomalies in the wireless body area networks which are used in real time healthcare systems. By identify anomalies, they decrease the false alarm rate which results in increasing the reliability of the system. They decide the size of the window by comparing the variance of predecessor and current sliding window. Pérez-Solano et al. [34] use adaptive window size for linear regression to synchronize time in wireless networks. They search for a window size which yields minimum Mean Square Prediction Error (MSPE). Similarly, Noor et al. [35] use the adaptive sliding window for signal segmentation which is used to recognize the transitional physical activity. The proposed method uses a default window size as a starting point and then used probability density function to expand the size of the window to capture the transitional activity with a longer duration. They keep increasing the size of the window iteratively until the probability density function reached its highest value.

Another work by Deypir et al. [36] use variable size sliding window to mine frequent itemsets. They start with an initial window size which is set by luser and then the window size is adjusted according to the rate of change in the incoming data stream. They increase the window size if no significant difference is detected and reduce the size if substantial changes are observed in the data. However, the limitation of this approach is that the size of the window becomes huge when there is no changes occurred in data. A recent work by Chou et al. [37] proposed a time series prediction system based on metaheuristic optimization of sliding windows. The purpose of the system is to forecast the stock price. However, their proposed method is computationally expensive and not feasible for real-time resource estimation.

Most of the existing works for resource estimation either use fixed observation windows or use simple statistical methods to decide the dynamic observation window sizes. However, some contributions use extensive search-based methods to identify the observation window sizes which are computationally intensive and infeasible for real-time resource estimations. Our work reported in this paper uses a novel deep learning based window size estimation method to efficiently identify the best observation windows to train the regression models for estimating future resource utilization.

3. Proposed system overview

The proposed system architecture is illustrated in Fig. 3. Different steps are numbered and labeled to explain the flow of the system. The proposed system works in the following steps.

i. Historical resource utilization logs of the data center are divided into sliding windows of a k fixed size intervals. Each sliding window at time interval t is called an observation window. Our objective is to identify an appropriate size for the observation window to train a resource prediction model with minimal prediction error.
ii. For each sliding window W_i, the system identifies the optimal window size to predict the next interval's resource consumption with minimal prediction error. In this phase, the next interval's consumption data for each sliding window is known. The system performs a linear search to identify the window size with minimal prediction error. The system checks the observation window into $k-1$ sub windows, start from length 2 to k. Each sub-window is used to estimate the resource consumption, and the size of the sub-window with minimum prediction error is identified as optimal window size Φ_{W_i} for the corresponding sliding window W_i, where $2 \geq \Phi_{W_i} \leq k$.
iii. Each sliding window W_i and the identified corresponding optimal window size Φ_{W_i} is recorded as training data set to predict the window size for resource utilization estimation.

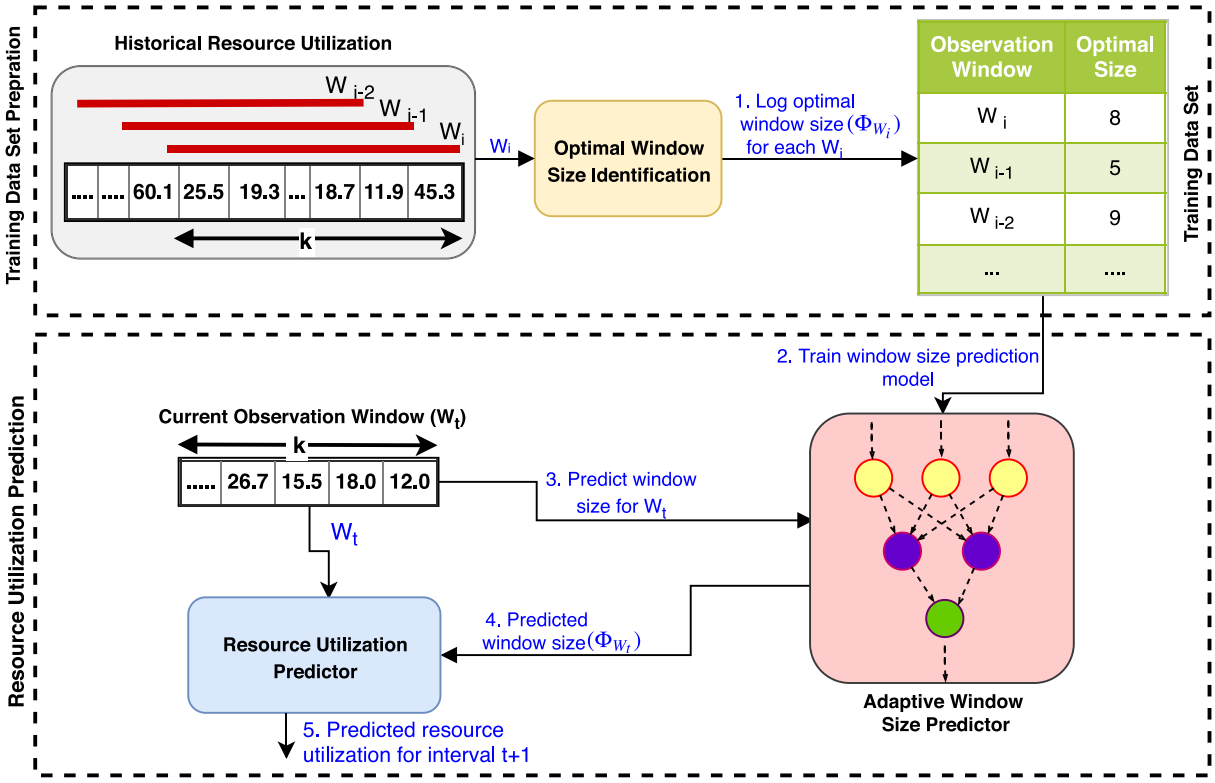


Fig. 3. Purposed system overview to learn adaptive window size predictor and using it to estimate the data center resource utilization.

iv. Once training data is prepared, the system train a deep neural network to predict the best window size for a given sliding window data. We named this component as "*Adaptive Window Size Predictor*".
v. Once *Adaptive Window Size Predictor* is trained then the system uses it to identify the best window size for each time interval t from the current observation window.
vi. The predicted window size is used to select the number of observations from the k data points of the current observation window for training a regression model to predict the resource utilization for the next time interval $t+1$.

In summary, the proposed system consists of "Adaptive Window Size Predictor" which predicts the number of most recent observations (window size) require to build estimation model for better estimation accuracy. This predictor is trained offline using historical resource utilization of all servers of a data center without machine identifications. The predicted window size from 'Adaptive Window Size Predictor' will be used as input to another machine learning algorithm to build resource estimation model for predicting next interval resource usage.

4. Adaptive window size predictor using deep learning

Predicting the adequate size of the observation window towards modeling the current analyzed series is crucial to saving computing power and also to improve the estimations when generating models periodically, but this prediction is not trivial. Here we propose to use Deep Learning (DL) based method to adaptively identify the observation windows.

DL is a set of machine learning methods based on neural networks, where those networks have more than one layer of transformations between the input data and the output data to be matched. MultiLayer Perceptrons (MLP), the proposed neural networks, consists on the transformation of an input data-set $X_{N,I}$ (a matrix of N observations and I features) to an output data-set $\hat{Y}_{N,J}$ (a matrix of J transformed features per observation).

A *Perceptron* (the usual neurons on MLPs) is an artifact that processes the input as $\hat{Y}_{N,I} = G(F(X_{N,I}))$, where $F(X_{N_I}) = X_{N,I} \cdot W_I + B$, and W_I is a matrix of weights to be adjusted, B is a vector of biases, and $G(X)$ is a function that in regression can be the Identity or in classification can be a *sigmoid* function. A typical "single hidden layer" MLP consists on an array of Perceptrons (the hidden layer) processing that input as $X'_{N,H} = G(F(X_{N,I}))$, where $F(X_{N,I}) = X_{N,I} \cdot W_{I,H} + B_H$, being H is the number of perceptrons on the hidden layer. The purpose of a layer is to find a non-linear relation between its inputs and outputs, then the results can be aggregated in the output layer as $\hat{Y}_{N,J} = G(F(X'_{N,H}))$. A deep MLP concatenates different hidden layers, and the output of a layer is the input of the next one. In that case, each layer i processes $\hat{X}^i_{N,H^i} = G(F(X^{i-1}_{N,H^{i-1}}))$, where H^i is the number of neurons in that layer. Training a MLP consists in finding the function MLP that transforms $MLP(X) = \hat{Y} \sim Y$, where Y is the real output data-set to be matched. Weight matrices W and bias vectors B are adjusted using Gradient Descent, by iteratively passing data through the network forth and back. The goal of adding multiple layers to a network is to learn latent patterns that a simple non-linear function cannot represent, through combinations of multiple non-linear relations.

While other approaches attempt shallow architectures, here we propose the use of deep neural networks, mainly because of the stochastic and non-linear nature of the workload data. As we tested in prior experiments with simpler models, simpler structures of neural networks have performed poorly when attempting to predict and discover the latent features in this kind of data. As seen in other works referring to resource utilization and Cloud and virtualization management, like [18,25,38], the use of deep neural networks on the prediction of resource usage and management has proven to be reasonably cost-efficient and

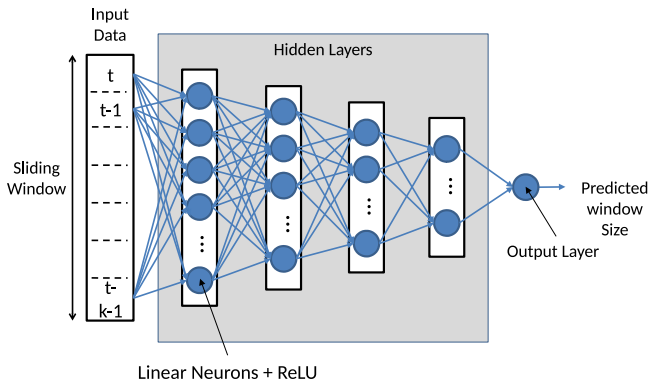


Fig. 4. Design of a 4-hidden layer Deep Neural Network which identifies adaptive observation window from a given time series of size k.

accurate, and this brings us to attempt these techniques for our adaptive window prediction scenario.

After testing different architectures and hyper-parameters, including the number of layers and number of neurons per layer, using a grid-search method, we selected the model resulting in the most accurate without ending with an overkill model. Such a model has an input of k elements as the size of the sliding window, and an output of 1 value, resulting in the prediction. The deep network has 4 hidden layers, with $\langle 23, 15, 10, 5\rangle$ hidden units each layer, with *rectified linear unit* (ReLU) activation function and normally random initialization. It is optimized using the *Adam* method for stochastic gradient descent [39], using the *Mean Squared Error* (MSE) as quality metric. The network has been trained for 500 epochs, with a batch size of 250 units, out of 11,8563 total elements on the training data-set, as the best tuning found on grid search. Fig. 4 shows the basic schema of our neural network.

The input data-set, being data a time series, is generated by sliding a window of size k, generating an observation for each window movement. The window is slide 1 time-step at a time. Each observation is introduced in the network as a vector of k values, considering that the window sample is always ordered. For validation purposes, the final data-set is randomly split 80/20 for training vs. testing subsets.

5. Estimation methods for resource utilization prediction

Our proposed technique to identify observation windows to build estimation method works with any machine learning and statistical estimations methods. We explore most commonly used machine learning methods for building resource estimation model and using the observation windows obtained through the proposed Deep Learning-based method. We explain the estimation methods in the following subsections.

5.1. Linear regression (LR)

Linear Regression (LR) is one of the simplest and widely used machine learning method for predictive modeling. LR assumes there is a linear relation between output variable y and input variables $X = \{x_1 \dots x_n\}$, and attempts to find a vector $\theta^T = \{\theta_1 \dots \theta_n\}$ and a scalar constant bias b where $\hat{Y} = X \cdot \theta + b$ while minimizing the error $\epsilon = |Y - \hat{Y}|$. Minimization is usually performed using the Least Squares Error approach and the cost function of LR for hypothesis θ for a given training examples X is calculated using:

$$\mathrm{J}(\theta) = \frac{1}{m}\sum_{i=1}^{m}(\theta^T \cdot X^{(i)} - Y^{(i)})^2. \tag{1}$$

In our proposed system, we use the predicted adaptive window size consisting of last m interval's resource estimation at time interval t to train LR model using cost function given in Eq. (1) and then estimate the resource estimation for $t+1$ time interval.

5.2. Ridge regression (RR)

Ridge regression (RR) also known as Tikhonov regularization is improved version of LR by introducing regularization to constraining the coefficients to low range. This helps to reduce the chances of model over-fitting. The cost function for Ridge regression for hypothesis θ is calculated using:

$$\mathrm{R}(\theta) = \frac{1}{m}\sum_{i=1}^{m}(\theta^T \cdot X^{(i)} - Y^{(i)})^2 + \alpha\frac{1}{2}\sum_{i=1}^{k}\theta_i{}^2, \tag{2}$$

where α is a hyperparameter use to control the regularization the model.

5.3. Least absolute shrinkage and selection operator (LASSO)

Least Absolute Shrinkage and Selection Operator (LASSO) is another improvement in LR by introducing regularization term and ensure to eliminates the least important features to increase the model accuracy. The cost function for LASSO for hypothesis θ is calculated using:

$$\mathrm{L}(\theta) = \frac{1}{m}\sum_{i=1}^{m}(\theta^T \cdot X^{(i)} - Y^{(i)})^2 + \alpha\sum_{i=1}^{k}\theta_i, \tag{3}$$

LASSO improves the prediction accuracy by selecting a subset of items rather than using all of them as compared to LR, NNLS, and Ridge regression which use all of the features and data.

5.4. Elastic net (EN)

EN improves LR using regulrization by combining Ridge and LASSO's regularizations. It also reduces the number of features by removing less important features to help improving accuracy of the model. The EN cost function is computed using:

$$\mathrm{E}(\theta) = \frac{1}{m}\sum_{i=1}^{m}(\theta^T \cdot X^{(i)} - Y^{(i)})^2 + \frac{1-r}{2}\alpha\sum_{i=1}^{k}\theta_i{}^2 + r\alpha\sum_{i=1}^{k}\theta_i, \tag{4}$$

where r is a mix ratio and can be control to include regularization of Ridge and LASSO. For example, $r = 1$ will result the EN to behave similar to LASSO and $r = 0$ will force the EN to behave similar to Ridge regression.

5.5. Non-negative least square (NNLS)

NNLS is a type of constrained least problems to restrict the coefficients of the model to positive numbers. This type of regression methods are feasible for resource estimation as the output is always positive. The objective function of NNLS is:

$$\underset{\theta_j \geq 0 \quad \forall j}{\arg\min}\sum_{i=1}^{m}(\theta^T \cdot X^{(i)} - Y^{(i)})^2. \tag{5}$$

The objective function (5) ensures that the linear coefficients in θ are non-negative. Since the resource usage of data centers are always non-negative, therefore all values in Y are also non-negative. As a result, we get non-negative prediction. We used the algorithm proposed by Lawson and Hansonb [40] to solve the NNLS objective function.

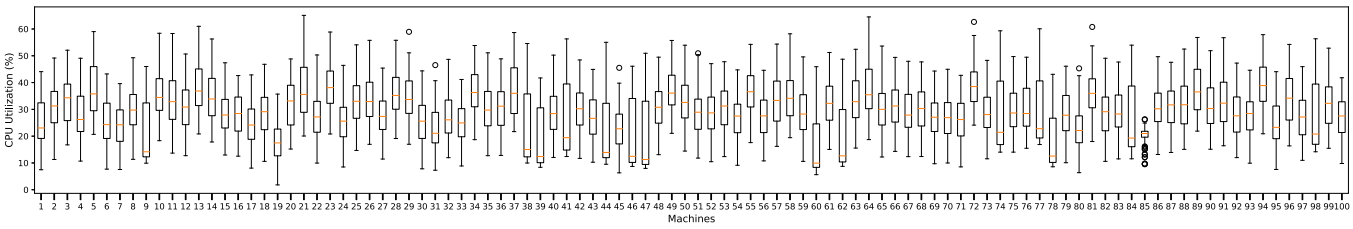


Fig. 5. CPU utilization for randomly selected 100 machines from Alibaba data set for twelve-hour data.

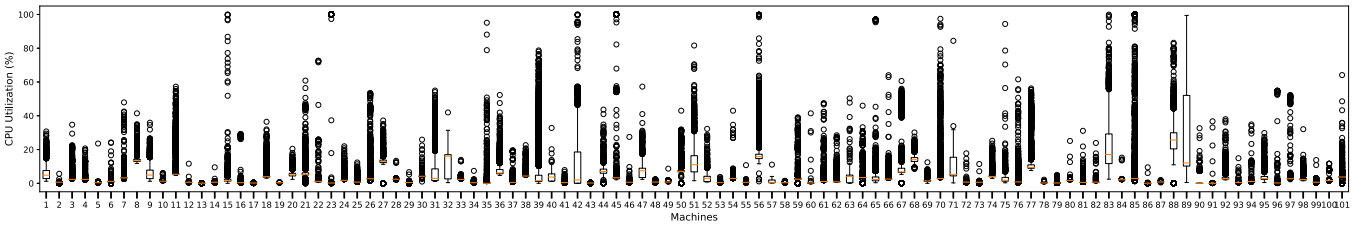


Fig. 6. CPU utilization for randomly selected 100 machines from Matenra data set for one-month data.

5.6. Support vector regression (SVR)

Support Vector Machine (SVM) methods are common for classification which maps the independent variable of a specific sample size into a high dimension feature space. Although SVM can be used for regression as Support Vector Regression machines (SVR) [41]. A SVR implementation follows similar principles to SVM but only differ to produce a continuous output variable. The compelling advantage of SVMs is to automatically learn non-linear functions as linear through transformation of data known as the *kernel trick*.

In a typical SVM implementation, input X are mapped into an h-dimensional feature space using a predefined non-linear kernel function to produce a linear model. Similar to LR, we can express SVMs as $\tilde{Y} = k(X) \cdot W + b$, where k is the function making the space for X linear. SMVs error minimization consists on building two margin functions (support vectors) $X \cdot W + b \pm \epsilon$, where final error ξ is computed for those elements outside the margins. As a disadvantage, margin ϵ can become an hyper-parameter.

6. Experimental setup and design

6.1. Data sets

To evaluate the proposed solution, we used three publicly available data sets representing diversified CPU resource utilization characteristics. Table 1 shows the total number of machines, average CPU load in percentage, average CPU load variations in the data sets. Each data set is categorized either low, moderate, or high CPU serving workloads. Materna data set is labeled as low CPU workload as the CPU load and variation is considerably low comparing to other data sets. Alibaba data set is considered as a traces of the data center serving moderate CPU workloads. Whereas, Bitbrains data set is recognized as a data center serving high CPU workloads. The data sets are briefly discussed in the following subsections.

6.1.1. Alibaba

Alibaba cluster logs [24] are publicly available data set containing performance traces of 1,313 machines for 12 h. The Alibaba cluster serves interactive and batch processing workloads. The available metrics in the data set are CPU, memory, and disk utilization for all machines representing average utilization for 5 min time intervals. Fig. 5 shows a CPU utilization sample for 100 randomly selected machines from the data set which shows that most of the machines CPU utilization remains from 20% to 50%. Therefore, we considered this data set representing moderate CPU workload serving data center.

Table 1
Data sets with CPU resource utilization statistics and utilization categories.

Data set	Total machines	Average load (%)	Average variation in load	Utilization category
Materna	520	4.44	9.29	Low
Alibaba	1,313	26.46	10.66	Moderate
Bitbrains	131	44.21	44.84	High

6.1.2. Materna

Materna is a service provider offering cloud services to the aviation industry. Their data center performance traces for 30 days are publicly available [42]. The data set contains resource utilization metrics for CPU, memory, network, and disk for 520 different VMs running on the data center. Each of these resource utilization metrics is sampled for 5 min average utilization. Fig. 6 shows CPU utilization for 100 randomly selected machines from the data set. Most of the machines CPU utilization remains below 5%; therefore, we considered this data set representing low CPU workload serving data center.

6.1.3. Bitbrains

Bitbrains is a service provider offer cloud services for enterprises. Their data center performance traces representing 1,750 VMs for 30 days is publicly available [43]. This data set also contains average CPU, memory, network, and disk utilization for all the VMs sampled for 5-minute interval. From this data set, we selected VMs with average CPU utilization greater than 30% (131 VMs) to build a data set representing data center serving high CPU workload. Fig. 7 shows a CPU utilization sample for 100 randomly selected machines from the data set which shows that most of the machines CPU utilization remains from 0% to 100%.

6.2. Evaluation criteria

To evaluate the proposed method, we used Mean Square Error (MSE) to quantify the error in resource estimation prediction. The MSE is calculated using true and estimated values using:

$$\text{MSE} = \frac{1}{n}\sum_{t=1}^{n}(a_t - p_t)^2, \tag{6}$$

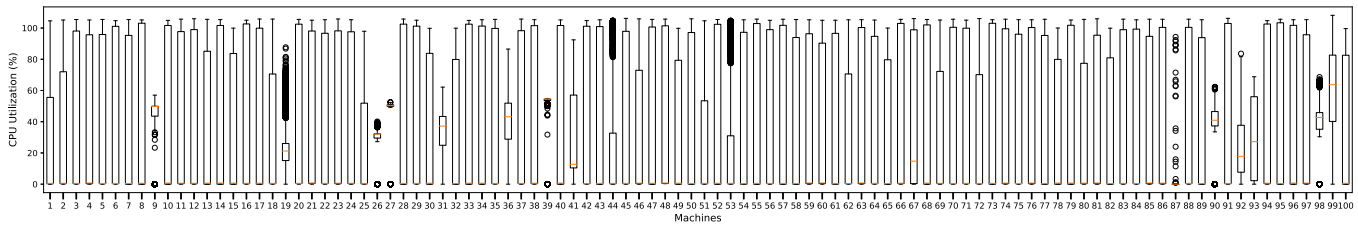


Fig. 7. CPU utilization for randomly selected 100 machines from Bitbrains data set for one-month data.

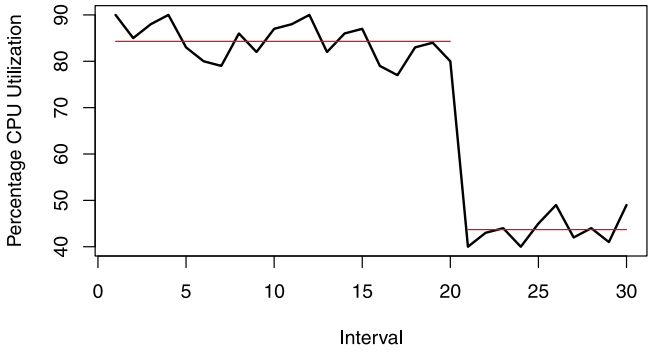


Fig. 8. Change Point Detection (CPD) method to identify observation window for building resource estimation model.

where a_t is the true CPU resource utilization and p_t is the estimated CPU utilization at tth time interval and n is the total number of estimations.

6.3. Experimental details

We performed four different experiments to evaluate and compare the proposed system. Experiment 1, 2 and 3 used to obtain results for using baseline methods whereas Experiment 4 is conducted using the proposed solution.

Specifically, in **Experiment 1 (FixW)**, we evaluate the effect of different fixed size observation windows on resource estimation accuracy for all three data sets. We use 5, 10, 20, and 30 fixed observation window sizes to train and estimate the resources using machine learning methods, explained in Section 5 for all three data sets.

In **Experiment 2 (CPD)**, we employ Change Point Detection (CPD) method [44] to adaptively identify the appropriate window size for training resource estimation models. This method allows selecting adaptive observations windows to build estimation models by using only data points after the recent change point. For example, Fig. 8 shows the use of CPD method to identify the adaptive observation window to use for training the estimation method. Assuming, the current time interval is 30 and CPD provides a change point at the 21st-time interval for the given data then we will use observations from 21st interval to the 30th interval for training an estimation model for future resource utilization estimation. The CPD method can identify multiple change points for the given time series data; however, we limit the observation window to the recent change point. In this experiment, we limit the maximum observation window size k to 30 intervals which represent the last 150 min of resource utilization observations.

In **Experiment 3 (Dalmazo)**, we use the dynamic window size method proposed by Dalmazo [26]. This method adjusts the window size by comparing the variance of the current and the last sliding window to predict the network traffic. We use their proposed method to dynamically identify the observation window for training and estimating the resource utilization for all three data sets. Similar to other baseline experiments, in this experiment, we also limit the maximum observation window size k to 30 intervals which represent the last 150 min of resource utilization observations.

In **Experiment 4 (Proposed)**, we use the proposed Adaptive Window Size Predictor (AWSP), explained in Section 4, to adaptively identify the best observation window to build resource estimation method. For every time interval, we provide the last 30 interval observations to the pre-trained AWSP and get the predicted window size and use the data for the predicted window size for building build an estimation model to predict the future resource utilization. In this experiment, we also limit the maximum observation window size k to 30 intervals which represent the last 150 min of the resource utilization observations.

7. Experiment results

7.1. Experiment 1: Fixed observation window (FixW)

Table 2 shows MSE for all three test sets using different resource estimation methods and fixed window sizes 5, 10, 20, and 30. MSE represents resource estimation error, and small values of MSE show better estimation results. We observed that small window sizes like 5 and 10 yield minimum MSE. For Alibaba data set, LR and RR yield minimum MSE using window size 10 whereas SVR, EN, LASSO, and NNLS produce minimum estimation error using window size 5. For Materna data set, LR, LASSO, RR, and NNLS give minimum MSE to estimate the CPU resource utilization whereas SVR and EN using window size 10 give minimum resource estimation error. For Bitbrain data set, window size 5 yield minimum error for all estimation methods. Overall EN with window size 5 outperforms all other estimation methods for all three data sets.

To compare different estimation methods and window sizes for all three data sets, we compute normalized MSE. Fig. 9 shows the normalized MSE for the results obtained in Experiment 1. EN for window size 5 gives minimum estimation error for all data sets whereas SVM yields worst results in all of the data sets for using window sizes 20 and 30. In general, the small window sizes with linear models show better results and exhibit that the resource utilization of data centers is locally linear. However, identifying the appropriate fixed window size to minimize the resource estimation error is a tedious and challenging task.

7.2. Experiment 2: Adaptive windows size using change point detection method

Table 3 shows the MSE for all three test data sets and different estimation methods using observation windows selected through the Change Point Detection (CPD) method [44]. The CPD selects adaptive observation windows to build estimation models by using data points after the recent change point from the given historical observations. SVR yields the minimum MSE for Alibaba and Materna data sets whereas EN produces the minimum MSE for Bitbrains data set using the CPD method.

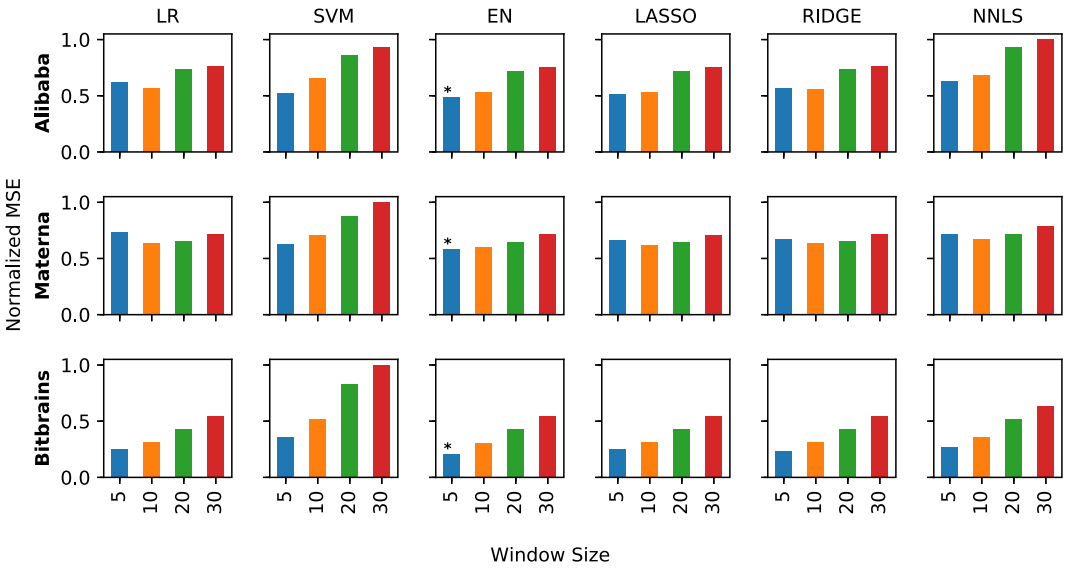


Fig. 9. Experiment 1 results showing Normalized MSE for comparison of different estimation methods and window sizes for all three data sets.

Table 2
Experiment 1 (FixW) results showing MSE for different estimation methods and fixed window sizes.

Data set	Window size	LR	SVR	EN	LASSO	RR	NNLS
Alibaba	30	34.28	42.15	34.06	34.02	34.28	45.25
	20	33.18	39.04	32.43	32.23	33.16	42.14
	10	**25.45**	29.5	23.92	24.09	**25.24**	30.91
	5	28	**23.72**	**22**	**23.28**	25.71	**28.49**
Materna	30	12.81	17.83	12.70	12.68	12.80	14.12
	20	11.63	15.69	11.45	11.45	11.62	12.85
	10	**11.39**	12.57	10.75	**10.95**	**11.29**	**11.98**
	5	13.02	**11.12**	**10.41**	11.81	11.91	12.70
Bitbrain	30	380.75	696.47	378.41	379.67	380.63	442.22
	20	301.83	577.35	297.75	300.46	301.47	360.44
	10	220.90	359.98	209.52	218.43	218.57	249.85
	5	**178.14**	**249.14**	**146.67**	**173.27**	**162.01**	**190.12**

Table 3
Experiment 2 (CPD) results showing MSE for different estimation methods.

Data set	LR	SVR	EN	LASSO	RR	NNLS
Alibaba	30.40	**23.24**	25.84	26.92	27.48	28.50
Materna	20.71	**11.02**	12.35	17.65	13.36	20.44
Bitbrains	272.05	292.55	**222.73**	262.29	234.68	276.49

Fig. 10 shows the adaptive window sizes obtained through the CPD method for building estimation models for first 100 test intervals of Alibaba, Materna, and Bitbrains data sets. At each test interval, a maximum of 30 previous observations are passed to the CPD and dynamically limit the observation windows to train the estimation models. For Materna data set, the CPD method does not variate the observation window so frequently whereas for Alibaba and Bitbrains the observation windows are frequently variates. This shows that for low variation data set like Materna the CPD methods also gives the observation windows with low variations. To further study the adaptive observation windows identified by the CPD for all three data sets, we draw the box plot.

Fig. 11 shows the box plot of the adaptive window sizes for the entire test sets of all three data sets. For Alibaba data set, the observation windows variate between 9 and 22 with 15 mean window size. For Bitbrains data set, in average the observation window size remains 30 and variate between 15 to 30 sizes. Whereas, Materna which represents low utilization workload with little variations in resource utilization pattern mostly use

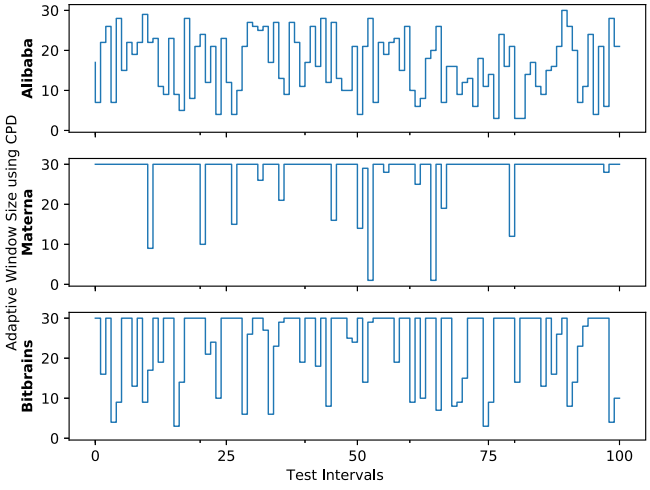


Fig. 10. Adaptive window sizes obtained using the CPD method in Experiment 2 for first 100 test intervals of all three data sets.

30 observation window size and show few outliers varying from 3 to 29. This clearly shows the limitation of the CPD method for the cases where resource utilizations do not show any notable changes in the usage pattern, and the CPD favors a bigger size observation windows.

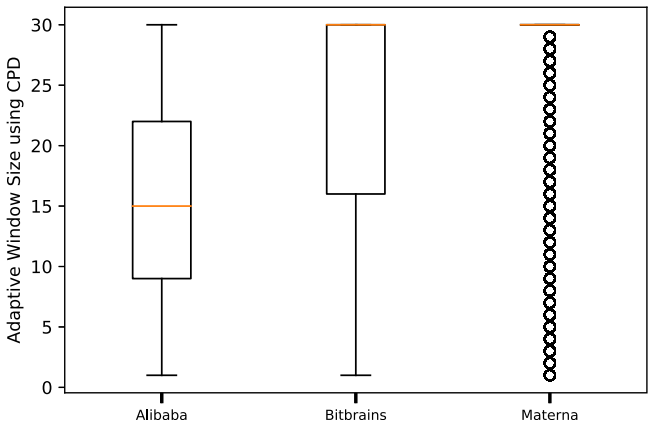


Fig. 11. Box plot of adaptive observation window sizes using the CPD method (Experiment 2) for test data sets.

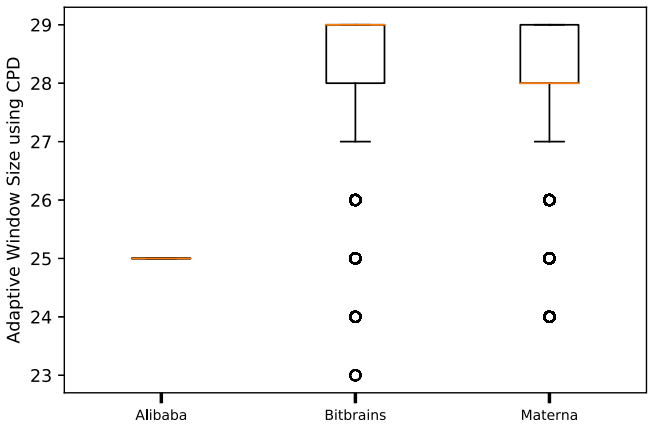


Fig. 12. Box plot of adaptive observation window sizes using the Dalmazo method (Experiment 3) for test data sets.

Table 4
Experiment 3 (Dalmazo) results showing MSE for different estimation methods.

Data set	LR	SVR	EN	LASSO	RR	NNLS
Alibaba	34.52	38.06	34.10	**33.96**	34.51	41.63
Materna	12.61	15.58	12.51	**12.49**	12.61	13.44
Bitbrains	432.57	777.72	**429.64**	431.16	432.41	494.70

Table 5
Experiment 4 (Proposed) results showing MSE for different estimation methods using the Proposed method to identify the observation window sizes.

Data set	LR	SVM	EN	LASSO	RR	NNLS
Alibaba	16.86	15.82	**14.97**	16.13	15.59	17.04
Bitbrains	154.70	169.21	**139.69**	156.38	148.08	156.08
Materna	10.19	10.11	9.45	9.63	9.93	**8.73**

The CPD method eliminates the need to search for the appropriate observation window size for building estimation method. However, the results obtained using the CPD methods are not outperforming FixW method results.

7.3. Experiment 3: Adaptive windows size using dalmazo [26] method

Table 4 shows the MSE for all three test data sets and different estimation methods using observation windows selected through the Dalmazo method [26]. LASSO yields the minimum MSE for Alibaba and Materna data sets whereas EN produces the minimum MSE for Bitbrains data set using the Dalmazo method.

Fig. 12 shows the box plot of the adaptive window sizes for the entire test sets of all three data sets. For Alibaba data set, the observation windows remain unchanged. For Bitbrains data set, in average the observation window size remains 29 and variate between 23 to 29 sizes. Whereas, Materna, which represents low utilization workload with little variations in resource utilization pattern mostly use 28 observation window size and show few outliers varying from 24 to 29. We found that most of the time, Dalmazo [26]'s method did not change the size of the observation window. This method selects window size based on the variance of the last sliding window and the current sliding window. The data sets consist of CPU resource utilization which ranges from 0 to 100, and the variance does not change significantly for two consecutive windows. Therefore most of the time, the window size remains unchanged. This clearly shows the limitation of the Dalmazo method for data center resource utilization's data sets.

7.4. Experiment 4: Adaptive windows using proposed method

Table 5 shows the MSE for all three test data sets and different estimation methods using the observation windows obtained through the proposed *Adaptive Window Size Predictor* method. For each data sets, all estimation methods yield comparable resource estimation results. However, EN gives the minimum MSE for Alibaba and Bitbrains data sets whereas NNLS produces the minimum MSE for Materna data set using the proposed method. Fig. 13 shows the adaptive window sizes obtained using the Proposed method for training estimation models for first 100 test intervals of three representative test data sets. For all data sets, the proposed method identifies different observation window sizes even for Materna data set which represents the workload with fewer variations. The proposed method variates the adaptive observation window sizes and yields better estimation results comparing to the CPD, Dalmazo and FixW methods.

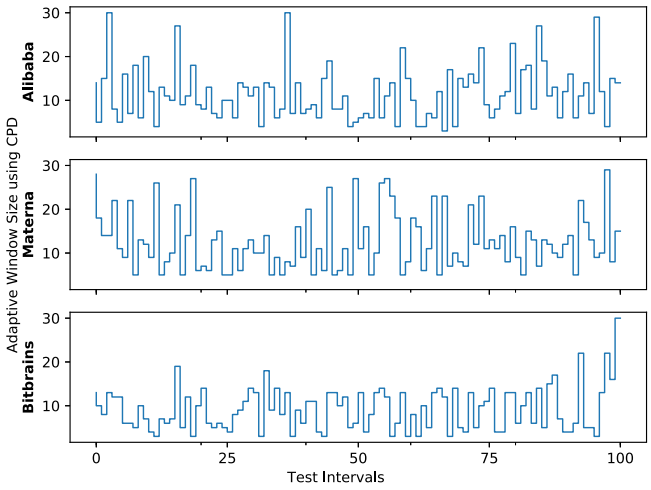


Fig. 13. Adaptive observation window sizes obtained using the Proposed method in Experiment 4 for first 100 test intervals of all three data sets.

Fig. 14 shows box plot for the adaptive window sizes identified by the proposed method for all three test data sets. For Alibaba data set, the observation window sizes variate between 2 to 30 with 13.26 mean. For Bitbrains data set, the observation window sizes variate between 2 to 30 with 9.65 mean and Materna data set shows the observation window sizes between 5 and 30 with 12.77 mean. The Proposed method outperforms the FixW, CPD and Dalmzao observation window selection methods and also helps to eliminate the daunting task for searching appropriate fixed window sizes.

Table 6
Normalized MSE representing resource estimation error on test data for Experiment 1 (FixW), Experiment 2 (CPD), Experiment-3 (Dalmazo) and Experiment-4 (Proposed).

Data set	Experimnet	LR	SVM	EN	LASSO	RR	NNLS	Average
Alibaba	Fix W	0.61	0.57	0.53	0.56	0.61	0.68	0.59
	CPD	0.73	0.56	0.62	0.65	0.66	0.68	0.65
	Dalmazo	0.83	0.91	0.82	0.82	0.83	1.00	0.87
	Proposed	**0.40**	**0.38**	**0.36**	**0.39**	**0.37**	**0.41**	**0.39**
Materna	Fix W	0.55	0.54	0.50	0.53	0.55	0.58	0.54
	CPD	1.00	0.53	0.60	0.85	0.64	0.99	0.77
	Dalmazo	0.61	0.75	0.60	0.60	0.61	0.65	0.64
	Proposed	**0.49**	**0.49**	**0.46**	**0.47**	**0.48**	**0.42**	**0.47**
Bitbrains	Fix W	0.23	0.32	0.19	0.22	0.21	0.24	0.24
	CPD	0.35	0.38	0.29	0.34	0.30	0.36	0.33
	Dalmazo	0.56	1.00	0.55	0.55	0.56	0.64	0.64
	Proposed	**0.20**	**0.22**	**0.17**	**0.20**	**0.19**	**0.20**	**0.20**

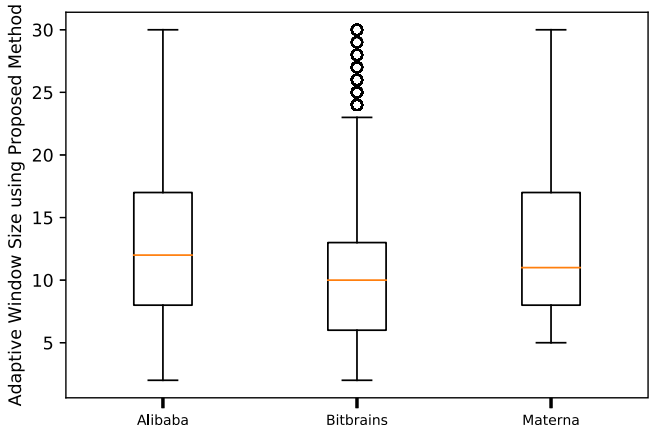


Fig. 14. Box plot of adaptive observation window sizes using the Proposed method (Experiment 4) for test data sets.

7.5. Comparison of FixW, CPD, Dalmazo and proposed method

In this section, we compare the results obtained in Experiment 1 (FixW), Experiment 2 (CPD), Experiment-3 (Dalmazo) and Experiment 4 (Proposed). Table 6 shows the normalized MSE for all four experiments and data sets. For each data sets, the maximum MSE among all four observation window selection methods and resource estimation is selected to calculate the normalized MSE. The proposed solution yields the minimum MSE using all estimation methods. The best performing estimation method is EN for Alibaba and Bitbrains data sets, whereas NNLS outperforms in Materna data set.

To quantify the relative improvement for estimation resource utilization using the Proposed window size selection method, we consider FixW (Experiment 1) as a baseline method. Fig. 15 shows the relative percentage of MSE for the Proposed, Dalmazo and CPD compared to the FixW. For all estimation methods, the Proposed outperforms the FixW, Dalmazo and the CPD to minimize the estimation error. Whereas, the FixW yields better estimation results compared to the CPD. For Alibaba data set, the proposed method produces 34%, 33,%, 32%, 31%, 38%, and 40% better estimation results for LR, SVM, EN, LASSO, RR, and NNLS

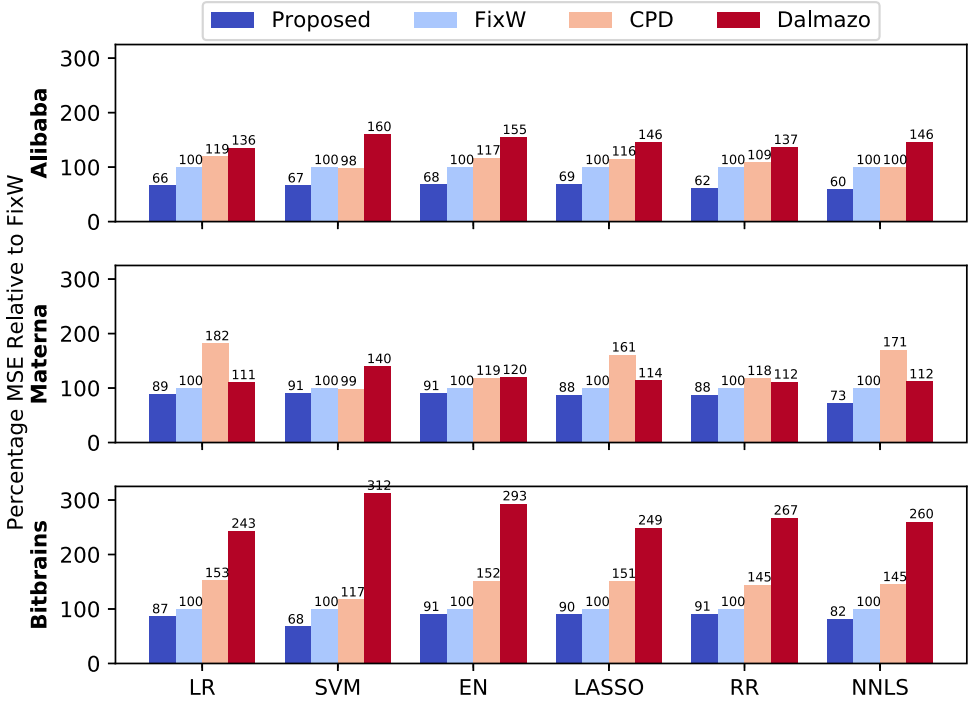


Fig. 15. Comparison of MSE for FixW, CPD, Dalmazo and Proposed method using different estimation methods.

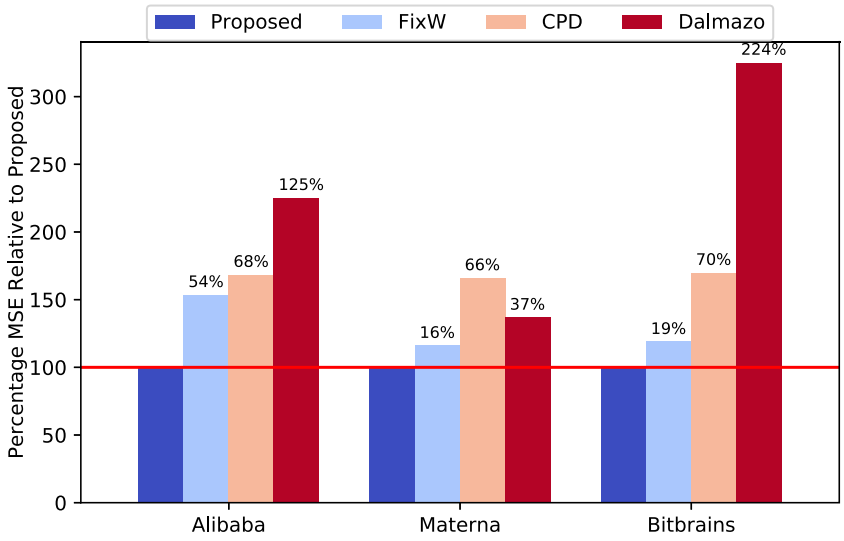


Fig. 16. Comparison of average relative percentage of MSE using different machine learning estimation methods for FixW, CPD and Dalmazo.

estimation methods respectively compared to FixW. For Materna data set, the Proposed method gives 11%, 9%, 9%, 18%, 18%, and 27% better estimation results for LR, SVM, EN, LASSO, RR, and NNLS estimation methods respectively compared to FixW. For Bitbrains data set, the Proposed method gives 13%, 32%, 9%, 10%, 9%, and 18% better estimation results for LR, SVM, EN, LASSO, RR, and NNLS estimation methods respectively compared to FixW.

Fig. 16 shows the average relative percentage of MSE using different machine learning estimation methods for the FixW, CPD, and Dalmazo compared to the proposed solution for Alibaba, Materna, and Bitbrains data sets. The comparison indicates that the Proposed method outperforms FixW, Dalmazo, and CPD for identifying appropriate observation window to build estimation models for better prediction accuracy. Our proposed solution yield 54%, 16%, and 19% less MSE for Alibaba, Materna, and Bitbrains data sets respectively over fix observation window (FixW) method. Whereas, Dalmazo and CPD give worst result than FixW for all data sets.

A traditional solution to identify observation windows for building estimation methods are based on a series of experiments using various fixed size observation windows manually and then determine the best observation window to use with the estimations. Whereas, an automatic adaptive observation window size can help to minimize estimating error better than fixed observation windows and also reduce the human efforts to test different observation window sizes manually. In our experimental evaluation, we evaluated six different commonly used estimation methods and showed that the proposed adaptive observation windows can yield better estimation results comparing to the fixed observation windows for all different estimation methods. We also show that our proposed adaptive observation windows identification methods perform better than Change Point Detection-based approach.

8. Conclusion

Data centers resource estimation is an active and challenging problem. The existing state-of-art methods available up-to-date use fixed size sliding windows to train estimation models. This traditional method does not consider the local changes and patterns in the resource usage of data centers and always using a fixed sized recent data points to use for training machine learning models which do not produce accurate future estimations.

In this paper, we devise a method to automatically limit the observation window size to use for building estimation models at every estimation step adaptively. Our proposed solution uses a multi-layer perceptron to identify the observation window sizes to be used by estimation methods for build models with improved estimation results to multiple baseline approaches. The proposed solution is evaluated on real resource utilization traces collected from Alibaba, Materna and Bitbrains data centers. Our proposed method can improve prediction accuracy from 16% to 54% over current methodologies. We conclude that the proposed system can help to identify the appropriate window size for each specific scenario over time for building estimation models with higher accuracy compared to the existing state-of-the-art baseline techniques. The proposed solution can be deployed on virtual machines or containers. However, the containers will provide native performance while virtual machines may introduce some overhead due to virtualization.

In the future, we intend to investigate the identification of the appropriate machine learning algorithm automatically for building estimation models using the proposed adaptive observation windows. Moreover, we will consider integrating statistical features including mean, variance, standard deviation, skewness with raw observations to increase the number of independent variables to improve the accuracy of the proposed model. We also intend to study the cost/performance trade-off for learning a set of machine-specific models over one model for the entire data center for identifying appropriate observation window size.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work is partially supported by the European Research Council (ERC) under the EU Horizon 2020 programme (GA 639595), the Spanish Ministry of Economy, Industry and Competitiveness (TIN2015-65316-P and IJCI2016-27485), the Generalitat de Catalunya, Spain (2014-SGR-1051) and University of the Punjab, Pakistan. The statements made herein are solely the responsibility of the authors.

References

[1] A.A. Rahmanian, M. Ghobaei-Arani, S. Tofighy, A learning automata-based ensemble resource usage prediction algorithm for cloud computing environment, Future Gener. Comput. Syst. 79 (2018) 54–71.

[2] K. Gai, M. Qiu, H. Zhao, X. Sun, Resource management in sustainable cyber-physical systems using heterogeneous cloud computing, IEEE Trans. Sustain. Comput. 3 (2) (2018) 60–72.

[3] C.G. Ralha, A.H. Mendes, L.A. Laranjeira, A.P. Araújo, A.C. Melo, Multiagent system for dynamic resource provisioning in cloud computing platforms, Future Gener. Comput. Syst. 94 (2019) 80–96.

[4] J. Li, Z. Lu, W. Zhang, J. Wu, H. Qiang, B. Li, P.C. Hung, SERAC3: SMart and economical resource allocation for big data clusters in community clouds, Future Gener. Comput. Syst. 85 (2018) 210–221.

[5] M. Ghobaei-Arani, S. Jabbehdari, M.A. Pourmina, An autonomic resource provisioning approach for service-based cloud applications: A hybrid approach, Future Gener. Comput. Syst. 78 (2018) 191–210.

[6] A. Hameed, A. Khoshkbarforoushha, R. Ranjan, P.P. Jayaraman, J. Kolodziej, P. Balaji, S. Zeadally, Q.M. Malluhi, N. Tziritas, A. Vishnu, et al., A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems, Computing 98 (7) (2016) 751–774.

[7] M. Dabbagh, B. Hamdaoui, M. Guizani, A. Rayes, Toward energy-efficient cloud computing: Prediction, consolidation, and overcommitment., IEEE Netw. 29 (2) (2015) 56–61.

[8] M. Xu, R. Buyya, Energy efficient scheduling of application components via brownout and approximate markov decision process, in: International Conference on Service-Oriented Computing, Springer, 2017, pp. 206–220.

[9] M. Xu, A.N. Toosi, R. Buyya, Ibrownout: An integrated approach for managing energy and brownout in container-based clouds, IEEE Trans. Sustain. Comput. 4 (1) (2018) 53–66.

[10] H. Duan, C. Chen, G. Min, Y. Wu, Energy-aware scheduling of virtual machines in heterogeneous cloud computing systems, Future Gener. Comput. Syst. 74 (2017) 142–150.

[11] M. Aldossary, K. Djemame, I. Alzamil, A. Kostopoulos, A. Dimakis, E. Agiatzidou, Energy-aware cost prediction and pricing of virtual machines in cloud computing environments, Future Gener. Comput. Syst. 93 (2019) 442–459.

[12] T.H. Nguyen, M. Di Francesco, A. Yla-Jaaski, Virtual machine consolidation with multiple usage prediction for energy-efficient cloud data centers, IEEE Trans. Serv. Comput. (2018) 1.

[13] M. Carvalho, D.A. Menascé, F. Brasileiro, Capacity planning for IaaS cloud providers offering multiple service classes, Future Gener. Comput. Syst. 77 (2017) 97–111.

[14] L. Tang, H. Chen, Joint pricing and capacity planning in the iaas cloud market, IEEE Trans. Cloud Comput. 5 (1) (2017) 57–70.

[15] W. Iqbal, M.N. Dailey, D. Carrera, P. Janecek, Adaptive resource provisioning for read intensive multi-tier applications in the cloud, Future Gener. Comput. Syst. 27 (6) (2011) 871–879.

[16] W. Wei, X. Fan, H. Song, X. Fan, J. Yang, Imperfect information dynamic stackelberg game based resource allocation using hidden Markov for cloud computing, IEEE Trans. Serv. Comput. 11 (1) (2018) 78–89.

[17] N. Kaur, S.K. Sood, An energy-efficient architecture for the Internet of Things (IoT), IEEE Syst. J. 11 (2) (2017) 796–805.

[18] Q. Zhang, L.T. Yang, Z. Yan, Z. Chen, P. Li, An efficient deep learning model to predict cloud workload for industry informatics, IEEE Trans. Ind. Inf. 14 (7) (2018) 3170–3178.

[19] S. Baig, W. Iqbal, J.L. Berral, A. Erradi, D. Carrera, Adaptive prediction models for data center resources utilization estimation, IEEE Trans. Netw. Serv. Manag. (2019) 1, http://dx.doi.org/10.1109/TNSM.2019.2932840.

[20] A.Y. Nikravesh, S.A. Ajila, C.-H. Lung, Towards an autonomic auto-scaling prediction system for cloud resource provisioning, in: Proceedings of the 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, IEEE Press, 2015, pp. 35–45.

[21] J. Yang, C. Liu, Y. Shang, B. Cheng, Z. Mao, C. Liu, L. Niu, J. Chen, A cost-aware auto-scaling approach using the workload prediction in service clouds, Inform. Syst. Front. 16 (1) (2014) 7–18.

[22] N.A. Davis, A. Rezgui, H. Soliman, S. Manzanares, M. Coates, FailureSim: A System for predicting hardware failures in cloud data centers using neural networks, in: Cloud Computing (CLOUD), 2017 IEEE 10th International Conference on, IEEE, 2017, pp. 544–551.

[23] Y. Ran, J. Yang, S. Zhang, H. Xi, Dynamic IaaS computing resource provisioning strategy with QoS constraint, IEEE Trans. Serv. Comput. 10 (2) (2017) 190–202.

[24] Alibaba Cluster log, https://github.com/alibaba/clusterdata.

[25] K. Mason, M. Duggan, E. Barrett, J. Duggan, E. Howley, Predicting host CPU utilization in the cloud using evolutionary neural networks, Future Gener. Comput. Syst. 86 (2018) 162–173.

[26] B.L. Dalmazo, J.P. Vilela, M. Curado, Online traffic prediction in the cloud: a dynamic window approach, in: Future Internet of Things and Cloud (FiCloud), 2014 International Conference on, IEEE, 2014, pp. 9–14.

[27] N. Wagner, Z. Michalewicz, An analysis of adaptive windowing for time series forecasting in dynamic environments: further tests of the DyFor GP model, in: Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation, ACM, 2008, pp. 1657–1664.

[28] K. Tschumitschew, F. Klawonn, Effects of drift and noise on the optimal sliding window size for data stream regression models, Comm. Statist. Theory Methods 46 (10) (2017) 5109–5132.

[29] Y. Wang, H. Li, A novel intelligent modeling framework integrating convolutional neural network with an adaptive time-series window and its application to industrial process operational optimization, Chemometrics Intel. Lab. Syst. 179 (2018) 64–72.

[30] T. Ouyang, X. Zha, L. Qin, A. Kusiak, Optimisation of time window size for wind power ramps prediction, IET Renew. Power Gener. 11 (8) (2016) 1270–1277.

[31] T. Ouyang, X. Zha, L. Qin, Y. Xiong, H. Huang, Model of selecting prediction window in ramps forecasting, Renew. Energy 108 (2017) 98–107.

[32] G. Smrithy, R. Balakrishnan, N. Sivakumar, Anomaly detection using dynamic sliding window in wireless body area networks, in: Data Science and Big Data Analytics, Springer, 2019, pp. 99–108.

[33] S.G.S. Nair, R. Balakrishnan, Mitigating false alarms using accumulator rule and dynamic sliding window in wireless body area, CSI Trans. ICT (2018) 1–6.

[34] J.J. Pérez-Solano, S. Felici-Castell, Adaptive time window linear regression algorithm for accurate time synchronization in wireless sensor networks, Ad Hoc Netw. 24 (2015) 92–108.

[35] M.H.M. Noor, Z. Salcic, I. Kevin, K. Wang, Adaptive sliding window segmentation for physical activity recognition using a single tri-axial accelerometer, Pervasive Mob. Comput. 38 (2017) 41–59.

[36] M. Deypir, M.H. Sadreddini, S. Hashemi, Towards a variable size sliding window model for frequent itemset mining over data streams, Comput. Ind. Eng. 63 (1) (2012) 161–172.

[37] J.-S. Chou, T.-K. Nguyen, Forward forecast of stock price using sliding-window metaheuristic-optimized machine-learning regression, IEEE Trans. Ind. Inf. 14 (7) (2018) 3132–3142.

[38] J.N. Witanto, H. Lim, M. Atiquzzaman, Adaptive selection of dynamic VM consolidation algorithm using neural network for cloud resource management, Future Gener. Comput. Syst. 87 (2018) 35–42.

[39] D.P. Kingma, J.L. Ba, Adam: A method for stochastic optimization, 2015, ArXiv preprint, arXiv:1412.6980.

[40] C.L. Lawson, R.J. Hanson, Solving Least Squares Problems, Siam, 1995.

[41] H. Drucker, C.J.C. Burges, L. Kaufman, A.J. Smola, V. Vapnik, Support vector regression machines, in: NIPS, MIT Press, 1996, pp. 155–161.

[42] Materna trace, http://gwa.ewi.tudelft.nl/datasets/gwa-t-13-materna.

[43] Bitbrains trace, http://gwa.ewi.tudelft.nl/datasets/gwa-t-12-bitbrains.

[44] G.J. Ross, et al., Parametric and nonparametric sequential change detection in R: The cpm package, J. Stat. Softw. 66 (3) (2015) 1–20.

Shuja-ur-Rehman Baig is currently pursuing the Ph.D. degree with Computer Architecture Department-UPC. He is also working as collaborator with the Data-Centric Computing group, Barcelona Supercomputing Center. He also holds a position of Lecturer at Punjab University College of Information Technology, University of the Punjab, Lahore, Pakistan. Shuja received M.Sc. degree in computer science from Lahore University of Management Sciences (LUMS), Lahore, Pakistan.

Waheed Iqbal is an Assistant Professor at Punjab University College of Information Technology, University of the Punjab, Lahore, Pakistan. He worked as a Postdoc researcher with the Department of Computer Science and Engineering, Qatar University during 2017–2018. His research interests lie in cloud computing, distributed systems, machine learning, and large scale system performance evaluation. Waheed received his Ph.D. degree from the Asian Institute of Technology, Thailand.

Josep Lluis Berral received the degree in informatics in 2007, the M.Sc. degree in computer architecture in 2008, and the Ph.D. degree from BarcelonaTech-UPC, computer science in 2013. He is a Data Scientist, working in applications of data mining and machine learning on data-centric computing scenarios, at the Barcelona Supercomputing Center. He was with the High Performance Computing Group, Computer Architecture Department-UPC, and the Relational Algorithms, Complexity and Learning Group, Computer Science Department-UPC.

David Carrera received the M.S. and Ph.D. degrees from BarcelonaTech-UPC in 2002 and 2008, respectively. He is an Associate Professor with the Computer Architecture Department, UPC. He is an Associate Researcher with the Data-Centric Computing, Barcelona Supercomputing Center. His research interests are focused on the performance management of data center workloads. He has been involved in several EU and industrial research projects. In 2015, he was awarded an ERC Starting Grant for the project HiEST. He was a recipient of the IBM Faculty Award in 2010.